A Design Technique for High-Performance Self-Checking Combinational Circuits

Elena Dubrova
Department of Electronics
Royal Institute of Technology
Kista, Sweden
elena@ele.kth.se

Abstract

In this paper, we present a new technique for design of the functional part of a self-checking combinational logic circuit, targeting high-performance applications. Our implementation is three-level AND-OR-AND logic, with the first two levels realized by PLAs and the third level realized by two-input AND gates. The outputs of the circuit are encoded in Berger code. Since such a design has inverters only on primary inputs, the polarity of the error produces on the output is the same as the polarity of the fault which caused this error. Therefore, the majority of realistic single faults and $2/3$ of the multiple stuck-at faults result in an unidirectional error on the output which are detected by Berger code. Experimental results show that, on average, an AND-OR-AND implementation with outputs encoded in Berger code is smaller (19%) than the non-encoded two-level PLA implementation of the same function.

1. Introduction

In the last decade, advances in integrated circuit technology have increased drastically the complexity of systems that can be realized as a ULSI scale single chip. This ongoing transition from traditional Application Specific Integrated Circuits to System-on-a-Chip has lead to new challenges and paradigm shifts in design methods, system and circuit architectures and testing techniques. Shrinking of device size and power supply levels, as well as increase in operating speed, result in reduction of noise margins. This makes the circuit increasingly sensitive to transient faults caused by single even upsets like atmospheric neutrons and alpha particles [11]. In order to maintain acceptable level of reliability, it is becoming mandatory to design ICs that are tolerant to these faults. Fault tolerant techniques using massive hardware redundancy, like duplication with comparison, are too costly to be used in commodity applications. Self-checking circuits, combining information and hardware redundancy, are an attractive alternative, allowing to achieve fault tolerance with a much lower hardware overhead.

In this paper we present a tool for automatic synthesis of the functional part of a self-checking circuit implemented in three-level AND-OR-AND logic. The first two levels are realized by PLAs and the third level is implemented by a set of two-input AND gates (Figure 1). Such an implementation has inverters only on primary inputs and therefore, for any internal line, all paths connecting the line to the circuit outputs have the same inversion parity. Therefore, the polarity of the error produces on the output is the same as the polarity of the fault which caused this error. If the outputs of such a circuit are encoded in an unidirectional error-detecting code, then this simple scheme allows detection of majority of single physical defects (including stuck-at faults, single-element faults (gate, transistor)), as well as a large part of multiple stuck-at faults (at least $2/3$, as we show later).

2. Self-checking circuits

For arbitrary combinational circuits the problem of choosing and designing a suitable code can be a very complex problem. Even a single stuck-at fault in the circuit can cause bidirectional errors in the output. Peterson and Rabin [6] show that the only code capable of detecting all errors in an arbitrary combinational circuit is straight duplication. It leads to a big area overhead, which is not acceptable for commodity applications. It is possible, however, to ensure by using certain design techniques that most of the single faults in the circuit cause only errors of a certain type. Two basic attributes which can be exploited to achieve the fault secure property are *maximum error multiplicity* and *error polarity*. Circuits of the first type are designed so that that

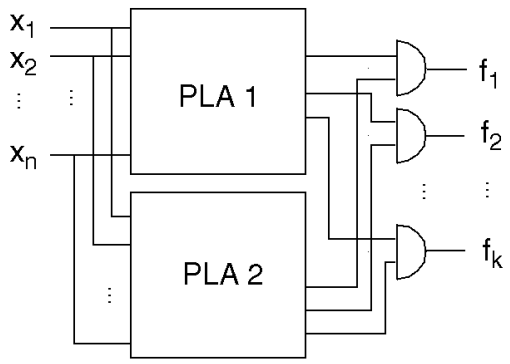

Figure 1. Three-level PLD implementing the functional part of the TSC circuit.

the number of output errors produced by a single fault in the circuit does not exceed a certain value t. Encoding the circuit's outputs in a t-error detection code allows detection of all singe faults [2], [9], [7]. Second type of the circuits are design to assure that, for any circuit line, all paths connecting the line to the circuit outputs have the same number of inverting gates modulo 2 (called *inversion parity* of the path). The inversion parity of a path determines the polarity of the error produces on the path output. For example, if the path inversion polarity is 0, then a fault $1 \rightarrow 0$ will cause the output error $1 \rightarrow 0$. Since the inversion parity is the same for *all* lines in the circuit, any number of multiple unidirectional faults will cause only unidirectional errors on the output. Encoding the outputs in a unidirectional code, such as Berger of m-of-n, allows detection of all multiple unidirectional errors [8]. Some methods use a combination of both approaches [5].

The technique we propose in this paper falls into the second category, exploiting the error polarity. The error polarity is related to the *unate* properties of the functions realized by the circuit. Internally unate circuits can be implemented by a Programmable Logic Array (PLA) [10] [5]. It is also possible to obtain a multi-level implementation of the circuits with inverters only on primary inputs, by apply De Morgan's laws to a multi-level circuit to pull the invertors to the inputs [4]. A PLA implementation is fast, but it might be quite large. A multiple-level implementation is usually smaller in area, but it is slower.

As a trade-off between these two, we present a design technique using a three-level Programmable Logic Device (PLD) consisting of PLA_1 and PLA_2, implementing the first two levels of logic, and a set of two-input AND gates, implementing the third level (Figure 1). Since PLAs produce the true and complement form of each variable as secondary inputs, such a PLD is unate with respect to the logical values carried by all internal lines of AND and OR arrays of PLAs as well as the the AND gates on the third level. Together with the inverters on the secondary inputs lines it is internally unate. Therefore any single stuck-at fault except the fault on primary inputs results in an unidirectional error on the output. Encoding the outputs of such a circuit in an unidirectional error-detecting code, like Berger, allows detection of all single stuck-at fault faults. Furthermore, it was proved in [11], that the circuits achieving fault secure property by means of output polarity (like in our case) also detect a more general class of *single-element faults*, in which only a single circuit element (line, gate, transistor) if affected by a fault, including timing faults and faults creating undetermined signal levels. Single-element faults are respected by the majority of physical defects, since most of the defects affect a single line, contact or transistor (opens, gate-oxide, gate-drain, source-drain shorts). The realistic faults outside this class are: some of the bridging faults (e.g. the ones creating undetermined values on the two bridged lines) and some faults caused by crosstalk [11].

Next, we analyze the case of multiple stuck-at faults. For a circuit with p lines (excluding the primary inputs), there are $3^p - 1$ multiple stuck-at-faults to be considered. All unidirectional (i.e. only stuck-at-0 or only stuck-at-1) multiple faults in the monotone part of the circuits cause a unidirectional error on the outputs of the circuit and are detected. We do not detect only those multiple faults, which include a line outside the monotone part of the circuit. There are as many such lines, as there are primary inputs. If the number of primary inputs are n, then the total number of the unidirectional multiple faults is $2/3 \cdot (3^{p-n})$, i.e. approximately 2/3 of the multiple faults for $n \ll p$. So, assuming that the primary inputs are faulty-free, the method guarantees to cover at least $2/3$ of the errors caused by multiple stuck-at faults.

3. Experimental results

We have developed a tool for automatic synthesis of the functional unit of a self-checking circuit. First, the output functions are encoded in a Berger code. Then, the algorithm for three-level minimization of AND-OR-AND expressions [3] is applied to map the encoded functions into a three-level PLD and to minimize its area. We have applied out tool to a set of benchmark functions to analyze the area overhead. Table 1 compares the number of products in the two-level AND-OR expression computed by Berkeley's two-level minimized Espresso [1] (column 3 for non-encoded outputs, p_E, and column 4 for encoded in Berger code outputs, p_{EB}), to the the number of products in the three-level AND-OR-AND expression computed by the algorithm [3] (column 5 for non-encoded outputs, p, and column 6 for encoded in Berger code outputs, p_B). Columns 2 and 3 give the number of inputs n and the number of out-

Table 1. Experimental results.

Example function	in. n	out. m	AND-OR p_E	AND-OR p_{EB}	AND-OR-AND p	AND-OR-AND p_B	$\frac{p^E}{p_B}$
5xp1	7	10	65	65	55	56	1.16
alu2	10	8	68	68	50	58	1.17
alu3	10	8	66	66	44	52	1.27
b9	16	5	119	340	119	190	0.63
b12	15	9	43	43	28	40	1.08
clip	9	5	120	135	120	100	1.20
dist	8	5	123	124	115	120	1.03
f51m	8	8	76	78	76	64	1.19
life	9	1	84	190	84	126	0.67
newtpla	15	5	23	44	19	41	0.56
newcpla1	9	16	38	45	30	37	1.03
radd	8	5	75	76	37	45	1.67
rd53	5	3	31	32	24	27	1.15
rd73	7	3	127	128	91	81	1.57
ryy6	16	1	112	119	7	30	3.73
sqn	7	3	38	49	33	49	0.78
t2	17	16	53	72	44	69	0.77
z4	7	4	59	60	29	39	1.51
Z5xp1	7	10	65	65	55	55	1.18
average							1.23

puts m of the benchmarks functions. Column 7 gives a comparison of the encoded AND-OR-AND versus non-encoded AND-OR implementation, computed as $\frac{p^E}{|g_1|+|g_2|}$.

The experimental results demonstrate that, on average, the AND-OR-AND implementation with outputs encoded in Berger code is 19% smaller than the non-encoded AND-OR PLA implementation of the same function. The number of products p_B in the encoded AND-OR-AND implementation can be up to 3 times smaller that the number of products p_E of the non-encoded AND-OR implementation (*ryy6*).

Note, that sometimes the number of products in the encoded AND-OR-AND expression is smaller than the number of products in the non-encoded AND-OR-AND expression. This paradox is due to the fact that the algorithm [3] is a heuristic and therefore the solution it finds is not always minimal. If the check-bit functions happen to be "suitable" for AND-OR-AND minimization, then the solution of the algorithm [3] can be smaller than the solution for the data-bit functions only.

4. Conclusions

We presented a new technique for synthesis of functional part of self-checking combinational logic circuit, targeting high-performance applications. Our implementation is three-level AND-OR-AND logic with the outputs encoded in Berger code. Such a design guarantees that the majority of realistic single faults and $2/3$ of the multiple stuck-at faults result are detected. Experimental results show that, on average, an encoded AND-OR-AND implementation is smaller than the non-encoded two-level PLA implementation of the same function.

References

[1] R.K. Brayton, G. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publisher, 1984.

[2] K. De, C. Natarajan, D. Nair, P. Banerjee, RSYN: A system for automated synthesis of reliable multilevel circuits, *IEEE trans. on VLSI Systems*, **2**, No 2, June 1994, pp. 184-195.

[3] E. Dubrova, P. Ellervee A fast algorithm for three-level logic optimization, *Proc. Int. Workshop on Logic Synthesis*, Lake Tahoe, May 1999, pp. 251-254.

[4] N. K. Jha and S-J. Wang, Design and synthesis of self-checking VLSI circuits, *IEEE Transactions on Computer-Aided Design and Systems*,**12**, June 1993, pp. 878–887.

[5] M. Nicolaidis, M. Boudjit, New implementations, tools and experiments for decreasing self-checking PLAs area overhead, *IEEE Int. Conf on Computer design*, Oct. 1991.

[6] W. W. Peterson and M. O. Rabin, On codes for checking logic operations, *IBM Journal of Research and Development*, **3**, 1959, pp. 163–168.

[7] V. V. Saposhnikov, VI. V. Saposhnikov, A. Morosov, M. Gossel, Design of self-checking unidirectional combinational circuits, *Proc. 7th Fault Tolerant Computing Symposium*, June 1997.

[8] J. E. Smith and G. Metze, The design of totally self-checking combinatorial circuits, *Proc. 7th Fault Tolerant Computing Symposium*, June 1997.

[9] N. A. Touba, E. J. McCluskey, Logic Synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection, *Proc. ICCAD*, 1994.

[10] X. Zhu, A. Breuer, A survey of testable PLA design, *IEEE Design and Test of Computers*, **5**, No 4, August 1988, pp.14-28.

[11] L. Znghel, M. Nicolaidis, I. Alzaher-Noufal. Self-checking circuits versus realistic faults in very deep submicron *VLSI Test Symposium*, pp 55-63, 2000.